# Problem Wk.1.4.4: OOPs

## Part 1: Thing 1

Below is a transcript of a session with the Python shell. This means that we are doing each of the statements in sequence, so for example, previous definitions and assignments are still in effect.

Provide the value of the expressions being evaluated and the type of the resulting value.

- If evaluating an expression would cause an error, select noneType and write `error` in the box.
- If the value is `None`, select noneType and enter `None`.
- If the value of an expression is a procedure or class, select the appropriate type and also write the name of the procedure or class in the box, as appropriate.
- If the value is an instance, write the Class name in the box.
- Select the appropriate type for integers, floats and lists and enter the value.

We encourage you to draw a diagram of the instances and their attribute values and update it as you work your way through the transcript.

1. 
```
>>> class Thing:
        def set(self, v):
            self.x = v
        def get(self):
            return self.x
>>> a = Thing()
>>> a.x = 6
>>> a.get()
```

   noneType
   int
   float
   boolean
   procedure
   class
   instance

2. 
```
>>> b = Thing()
>>> a.set(b)
>>> a.x
```

   noneType

3. 
```
>>> b.set(7)
>>> a.x.x
```

   noneType

4. 
```
>>> a.get()
```

   noneType

This problem contains one or more multiple-choice questions. When a given set of choices is used for the first time, all choices are displayed. If there are several related questions, you can assume that the same choices are available for all questions.

5. `>>> a.x.get()`

   noneType

6. `>>> 3 + a.get().get()`

   noneType

7. ```
   >>> c = a.get()
   >>> c.x
   ```

   noneType

8. ```
   >>> a.set(1 - a.get().get())
   >>> a.x
   ```

   noneType

9. ```
   >>> c.set(3)
   >>> a.get().get()
   ```

   noneType

10. ```
    >>> a = Thing()
    >>> b = Thing()
    >>> a.set(b)
    >>> b.set(a)
    >>> a.x == b
    ```

    noneType

11. `>>> a.x.x == a`

    noneType

12. `>>> a.x.x.x == b`

    noneType

---

## Part 2: Thing 2

Below is a transcript of a session with the Python shell. This means that we are doing each of the statements in sequence, so for example, previous definitions and assignments are still in effect.

Provide the value of the expressions being evaluated and the type of the resulting value.

- If evaluating an expression would cause an error, select noneType and write `error` in the box.
- If the value is `None`, select noneType and enter `None`.
- If the value of an expression is a procedure or class, select the appropriate type and also write the name of the procedure or class in the box, as appropriate.
- If the value is an instance, write the Class name in the box.
- Select the appropriate type for integers, floats and lists and enter the value.

1. ```
   >>> def thingMangle(arg):
           arg.set(arg.get() + 1)
           arg.hasBeenMangled = True
   >>> a = Thing()
   ```
   ```

```
>>> a.set(5)
>>> thingMangle(a)
>>> a.get()
```

noneType
int
float
boolean
procedure
class
instance

2. `>>> a.hasBeenMangled`

   noneType

3. ```
   >>> b = Thing()
   >>> b.set(Thing())
   >>> b.get().set(3)
   >>> thingMangle(b.get())
   >>> b.get()
   ```

   noneType

4. `>>> b.get().get()`

   noneType

5. ```
   >>> c = Thing()
   >>> thingMangle(c)
   ```

   noneType

---

## Part 3: Thing mangle

Add a method called `mangle` to the `Thing` class, which has the same effect as `thingMangle`. That is:

```
a = Thing()
a.set(3)
a.mangle()
```

should be equivalent to:

```
a = Thing()
a.set(3)
thingMangle(a)
```

Use the `set` and `get` methods of `Thing`, do not access `x` directly.

```
class Thing:
   def set(self, v):
      self.x = v
   def get(self):
      return self.x
```

---

## Part 4: More mangling

Define a procedure `mangled` that takes one argument, a number `z`, and which does:

- Creates a new `Thing`,
- set its `x` value to be `z`,
- mangles it, and
- returns it.

Use the `set`, `get` and `mangle` methods of `Thing`, do not access `x` directly.

```
def mangled(z):
   pass
```

MIT OpenCourseWare
http://ocw.mit.edu

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.